Yugoslav Journal of Operations Research
Vol 19 (2009), Number 1, 3-40
DOI: 10.2298/YUJOR0901003P

AN OVERVIEW ON POLYNOMIAL APPROXIMATION OF NP-HARD PROBLEMS

Vangelis Th. PASCHOS

LAMSADE, CNRS UMR 7024 and University Paris-Dauphine
Place du Maréchal de Lattre de Tassigny, France
paschos@lamsade.dauphine.fr

Received: December 2007 / Accepted: May 2009

Abstract: The fact that polynomial time algorithm is very unlikely to be devised for an optimal solving of the NP-hard problems strongly motivates both the researchers and the practitioners to try to solve such problems heuristically, by making a trade-off between computational time and solution's quality. In other words, heuristic computation consists of trying to find not the best solution but one solution which is "close to" the optimal one in reasonable time. Among the classes of heuristic methods for NP-hard problems, the polynomial approximation algorithms aim at solving a given NP-hard problem in polynomial time by computing feasible solutions that are, under some predefined criterion, as near to the optimal ones as possible. The polynomial approximation theory deals with the study of such algorithms. This survey first presents and analyzes time approximation algorithms for some classical examples of NP-hard problems. Secondly, it shows how classical notions and tools of complexity theory, such as polynomial reductions, can be matched with polynomial approximation in order to devise structural results for NP-hard optimization problems. Finally, it presents a quick description of what is commonly called inapproximability results. Such results provide limits on the approximability of the problems tackled.

Keywords: Computational complexity, approximation algorithms.

1. WHAT IS POLYNOMIAL APPROXIMATION AND WHY WE DO IT?

It is widely believed that no polynomial algorithm can be devised for an optimal solving of the NP-hard problems. So, several approaches, more or less satisfactory, can be followed when one tries to solve them. Roughly speaking, these approaches belong to one of the following two families.

Exact (optimal) algorithms, that compute optimal solutions for the problems but run in exponential time; such algorithms are based upon either search tree-based methods (branch-and-bound, branch-and-cut, branch-and-price, etc.), or upon dynamic programming etc.

Heuristic methods, that run faster than the exact algorithms and compute sub-optimal solutions (that are, sometimes, unfeasible); the most notorious among these methods being:

- the polyhedral methods,
- the metaheuristics (simulated annealing, genetic algorithms, tabou, etc.),
- the *polynomial approximation algorithms with (a priori) performance guarantees*.

The goal of this overview is to make a general presentation of the last category of heuristic methods, that is, the polynomial approximation algorithms.

What is polynomial approximation? Roughly speaking, this is the art to achieve feasible solutions with the objective value as close as possible (in some predefined sense) to the optimal value in polynomial time. How can we do it? Hopefully, the reader will see it in what follows.

Why do we use polynomial approximation rather than other heuristic methods? There are several reasons for that. The main reasons are, in our opinion, both "operational" and "structural".

A central operational reason is that there are problems that describe natural situations and either require *feasible* solutions to be obtained quickly (i.e., in polynomial time) and this fact must be guaranteed a priori, or where optimum is misdefined or senseless. Also, it is sometimes necessary for a decision maker to have an a priori estimation of the performance of the algorithms used in the context of handling these problems.

On the other hand, the main structural reason is that polynomial approximation is a natural extension of the complexity theory into the combinatorial optimization and it largely contributes to the enrichment of both these domains.

The aim of the study of polynomial approximation of combinatorial optimization problems is to characterize the ability of a specific problem to be "well-solved" in polynomial time. This is done by determining both the upper and the lower bounds of approximability, i.e., by exhibiting specific approximation algorithms that have achieved a given level of approximation on one side, and, on the other side, showing that no algorithm can possibly provide a better approximation level, until something strange and unexpected happens, i.e., until a very highly improbable complexity hypothesis (e.g., P = NP) holds.

The existence of polynomial approximation as a scientific area was started by a seminal paper by [46]. Since then, this research programme is one of the most active research programmes in operational research, combinatorial optimization and theoretical computer science.

2. PRELIMINARIES

The object of polynomial approximation is the class of the so-called NPO problems. Informally, this is the class of optimization problems the "decision"-counterparts of which are in NP. Let us recall that, for a maximization problem (the case

of a minimization problem is completely analogous), its decision version can be expressed as follows:

"Is there a solution with value *greater than, or equal to* K ?", where K is a constant that, for the decision version, is a part of the instance.

Solving a decision problem Π becomes to answer the question defining Π by *yes* or *no* correctly. For example, for the decision version of MAX INDEPENDENT SET[1]: "given a graph G and a constant K, is there an independent set of G of the size at least K ?", to solve it becomes to correctly answer the question if G has an independent set of size at least K or not.

Formally, an NPO problem Π is a four-tuple (***I***, Sol, *m*, goal) such that:

- ***I*** is the set of instances (recognizable in polynomial time);
- for I $\in$ ***I***, Sol(I) is the set of feasible solutions of I; feasibility of any solution can be decided on in polynomial time;
- for any I $\in$ ***I***, at least one feasible solution can be computed in polynomial time;
- the objective value $m(I,S)$ of any solution S, is computable in polynomial time;
- goal $\in \{\min, \max\}$.

We shall now briefly present some of the basic notions of the polynomial approximation theory. More details can be found in [5, 44, 66, 72].

Having given an instance I of a combinatorial maximization (resp., minimization) problem Π = (***I***, Sol, *m*, goal), we denote by $\omega(I)$, $m_{\mathrm{A}}(I,S)$ and $opt(I)$ the value of the worst solution of I (in the sense of objective function), the value of a solution S (computed by some polynomial time approximation algorithm A supposed to feasibly solve the problem Π), and the optimal value[2] for I, respectively. The worst solution of I is the optimal solution for I with respect to the NPO problem Π' = (***I***, Sol, *m*, goal′) where:

$$\text{goal}' = \begin{cases} \max & \text{if goal} = \min \\ \min & \text{if goal} = \max \end{cases}$$

There exist mainly two paradigms dealing with polynomial approximation.

Standard approximation. The quality of an approximation algorithm A is expressed by the ratio:

$$\rho_{\mathrm{A}}(I) = \frac{m_{\mathrm{A}}(I,S)}{opt(I)}$$

Note that the approximation ratio for minimization problems is in $[1,\infty)$, while for the maximization ones this ratio is in $(0,1]$.

Differential approximation. The quality of an approximation algorithm A is expressed by the ratio:

---

[1] We properly define this problem in Section 4.1.

[2] The value of an optimal solution.

$$\delta_{\mathrm{A}}(I) = \frac{|\omega(I) - m_{\mathrm{A}}(I,S)|}{|\omega(I) - opt(I)|}$$

The value of the differential ratio is always in $[0,1]$, independently on the optimization goal of the problem.

For both paradigms, the closer the ratios are to 1, the better the performance of the approximation algorithm A. Let us also note that, as we will see, the results obtained by adopting one or the other of the two paradigms are very often different even for the same problem.

The rest of the paper has been organized as follows. In Section 3 the main approximability classes have been defined. In Section 4 we have analyzed the polynomial approximation algorithms for several well-known hard optimization problems. In Section 5, we have shown how the main tool of the complexity theory, polynomial reductions, can be adapted to the framework of polynomial approximation, in order to produce structural results. In Section 6 we have said a few words about the limits of approximability, i.e., for inapproximability of NP-hard problems. Providing inapproximability results that state that a specific problem is not approximable within better than some approximation level is crucial (although somewhat far from the classical operational researchers concerns; for this reason this part of the paper will be short) for characterizing the approximability of NP-hard problems and for the understanding of their structure. Finally, in Section 7 we have presented a quick overview on the completeness result in the approximation classes.

To be brief, only the problems discussed in detail in this paper will be defined. For the other ones, the interested reader can referred to [35]. Also, for the notions and definitions of the graph theory, one can refered to [16].

3. APPROXIMATION CLASSES

According to the best approximation ratios known for them, NP-hard problems are classified into *approximability classes*. These classes create a kind of hierarchy in the class of the NP-hard problems. The best known among them (going from the pessimistic to the optimistic ones) are the following classes (for any standard-approximation class C, DC denotes the respective differential class).

Exp-APX and Exp-DAPX. The classes of problems for which the best ratio known is exponential (or the inverse[3] of an exponential) with the size of their instance. The notorious member of Exp-APX is MIN TSP. On the other hand, no natural combinatorial optimization problem is still known to be in Exp-DAPX.

Poly-APX and Poly-DAPX. The classes of problems for which the best ratio known is polynomial (or the inverse of a polynomial) with the size of their instance. MAX INDEPENDENT SET, MAX CLIQUE, MIN COLORING, etc., belong to Poly-APX. On the other hand, MAX INDEPENDENT SET, MAX CLIQUE, MIN VERTEX COVER, MIN SET COVER, etc., belong to Poly-DAPX.

Log-APX and Log-DAPX. The classes of problems for which the best ratio known is an logarithm (or the inverse of an logarithm) of the size of their instance. MIN

[3] Recall that when goal=max the approximation ratio is smaller than 1.

SET COVER and MIN DOMINATING SET are the most notorious representatives of Log-APX. On the other hand, no natural combinatorial problem is known to be in Log-DAPX.

APX and DAPX. Here the "more optimistic" approximability classes start. APX and DAPX are the classes of problems approximable within the ratios that are fixed constants. MIN VERTEX COVER, MIN METRIC TSP[4], BIN PACKING, MAX TSP, etc., belong to APX, while MIN TSP, MAX TSP, MIN COLORING, etc., belong to DAPX.

PTAS and DPTAS. The classes of problems admitting *polynomial time approximation schemata*. A polynomial time approximation scheme, is a sequence of algorithms A_ε achieving ratio $1+\varepsilon$, for every $\varepsilon > 0$ ($1-\varepsilon$ for maximization problems, or for the differential paradigm), in time which is polynomial with the size of the instance but exponential with $1/\varepsilon$. MAX PLANAR INDEPENDENT SET[5], MIN PLANAR VERTEX COVER[6], MIN EUCLIDEAN TSP[7], etc., are in PTAS. On the other hand, MAX INDEPENDENT SET, MIN PLANAR VERTEX COVER, BIN PACKING, etc., are known to belong to DPTAS.

FPTAS and DFPTAS. The classes of problems admitting *fully polynomial time approximation schemata*. A fully polynomial time approximation scheme is a polynomial time approximation scheme that is, furthermore, polynomial with $1/\varepsilon$. Also, KNAPSACK is in both FPTAS and DFPTAS.

Let us also mention the existence of another approximability class denoted by 0-DAPX (defined in [13]) that is meaningful only for the differential approximation paradigm. 0-DAPX is the class of problems for which any polynomial algorithm returns the worst solution in at least one of their instances. In other words, for problems in 0-DAPX, the differential approximation ratio is ***equal*** to 0. MIN INDEPENDENT DOMINATING SET is known to be in 0-DAPX.

Finally, let us note that, judging by the way the approximability classes are defined (i.e., as functions of the instance size), there is indeed a continuum of such classes. Figures 1 and 2 illustrate the approximability classes landscapes for standard and differential approximability paradigms, respectively.

Dealing with the classes defined above, the following inclusions hold:

$$\text{PO} \subset \text{FPTAS} \subset \text{PTAS} \subset \text{APX} \subset \text{Log-APX} \subset \text{Poly-APX} \subset \text{Exp-APX} \subset \text{NPO}$$
$$\text{PO} \subset \text{DFPTAS} \subset \text{DPTAS} \subset \text{DAPX} \subset \text{Log-DAPX} \subset \text{Poly-DAPX} \subset \text{Exp-DAPX}$$
$$\subset \text{0-DAPX} \subset \text{NPO}$$

These inclusions are strict unless P = NP. Indeed, for any of these classes, there are natural problems that belong to each of them but not to the immediately smaller ones. For instance, for the standard paradigm:

---

[4] MIN TSP in complete graphs whose edge-weights verify the triangle-inequality.

[5] MAX INDEPENDENT SET in planar graphs.

[6] MIN VERTEX COVER in planar graphs.

[7] MIN METRIC TSP in (0,1)-plane.

$$\begin{aligned}
\text{KNAPSACK} &\in \text{FPTAS} \setminus \text{PO}\\
\text{MAX PLANAR INDEPENDENT SET} &\in \text{PTAS} \setminus \text{FPTAS}\\
\text{MIN VERTEX COVER} &\in \text{APX} \setminus \text{PTAS}\\
\text{MIN SET COVER} &\in \text{Log-APX} \setminus \text{APX}\\
\text{MAX INDEPENDENT SET} &\in \text{Poly-APX} \setminus \text{Log-APX}\\
\text{MIN TSP} &\in \text{Exp-APX} \setminus \text{Poly-APX}
\end{aligned}$$

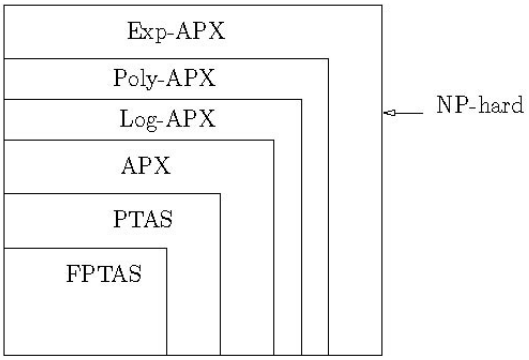

Figure 1: Standard approximability classes (under the assumption P ≠ NP).

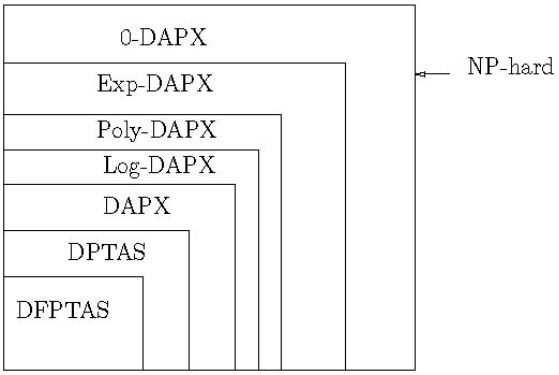

Figure 2: Differential classes (under the assumption P ≠ NP).

4 APPROXIMATION ALGORITHMS FOR SEVERAL NP-HARD PROBLEMS

4.1. Max independent set

Given a graph $G(V,E)$, the MAX INDEPENDENT SET problem consists of determining a maximum-size set $V' \subseteq V$ so that, for every $(u,v) \in V' \times V'$, $(u,v) \notin E$.

Let us first consider the integer-linear formulation of the MAX INDEPENDENT SET and its linear relaxation, denoted by MAX INDEPENDENT SET-R, where in the given graph $G(V,E)$, A denotes its incidence matrix:

$$\text{MAX INDEPENDENT SET} = \begin{cases} \max & \vec{1} \cdot \vec{x} \\ & A\vec{x} \leqslant \vec{1} \\ & \vec{x} \in \{0,1\}^n \end{cases} \quad (1)$$

$$\text{MAX INDEPENDENT SET-R} = \begin{cases} \max & \vec{1} \cdot \vec{x} \\ & A\vec{x} \leqslant \vec{1} \\ & \vec{x} \in [0,1]^n \end{cases} \quad (2)$$

The following seminal theorem, due to [57] gives a very interesting characterization for the basic optimal solution of (2).

Theorem 1. *([57]). The basic optimal solution of MAX INDEPENDENT SET-R is semi-integral, i.e., it assigns values from* $\{0,1,1/2\}$ *to the variables. If* V_0, V_1 *and* $V_{1/2}$ *are the subsets of* V *associated with 0, 1 et 1/2, respectively, then there exists a maximum independent set* S^* *so that:*

1. $V_1 \subseteq S^*$
2. $V_0 \subseteq V \setminus S^*$

A basic corollary of Theorem 1 is that in order to solve MAX INDEPENDENT SET, one can first solve its linear relaxation MAX INDEPENDENT SET-R (this can be done in polynomial time [3, 37, 48, 69]) and store V_1 and then solve the MAX INDEPENDENT SET in some way in $G[V_{1/2}]$, i.e., the subgraph of G induced by $V_{1/2}$.

Indeed, the solution of the MAX INDEPENDENT SET-R provides sets V_0, V_1 and $V_{1/2}$ that form a partition of V. Furthermore, by the constraint set of this program, edges can exist into V_0 and $V_{1/2}$ and between V_1 and V_0 and V_0 and $V_{1/2}$, but not between V_1 and $V_{1/2}$ (see also Figure 3 where thick lines indicate the possible existence of edges between vertex-sets). So, the union of V_1 (that is an independent set per se) and of an independent set of $G[V_{1/2}]$ is an independent set for the whole of G.

Let us now consider the following algorithm, due to [43], denoted by IS:

1. solve MAX INDEPENDENT SET-R in order to determine V_0, V_1 et $V_{1/2}$;

2. color $G[V_{1/2}]$ with at most $\Delta(G[V_{1/2}])$ colors (where, for a graph G, $\Delta(G)$ denotes its maximum degree) using the algorithm by [53]; let $\hat{S}$ be the largest color
3. output $S = V_1 \cup \hat{S}$

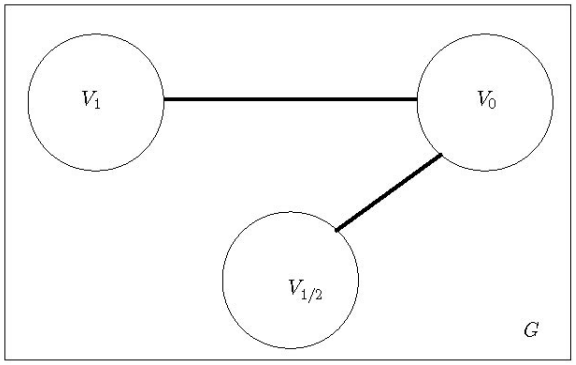

Figure: 3: A graph G and the possibility of existence of edges between sets V_0, V_1 and $V_{1/2}$ computed by the solution of MAX INDEPENDENT SET-R.

Theorem 2. *The algorithm IS achieves the approximation ratio* $2/\Delta(G)$ *for the MAX INDEPENDENT SET problem.*

Proof. Let us first recall that the vertices of a graph G can be feasibly colored[8] with at most $\Delta(G)$ colors in polynomial time ([53]). This result is, in fact, the constructive proof of an existential theorem about such coloring originally stated by [18].

Fix a maximum independent set S^* of G that contains V_1 (by item 1 in Theorem 1 this is always possible). Since $\hat{S}$ is the largest of the at most $\Delta(G[V_{1/2}])$ colors (independent sets) produced in step 2 of Algorithm IS, its size satisfies:

$$\left|\hat{S}\right| \geq \frac{\left|V_{1/2}\right|}{\Delta\left(G\left[V_{1/2}\right]\right)} \tag{3}$$

The size of S returned by the algorithm at step 3 is given by:

$$m(S,G) = |S| = |V_1| + \left|\hat{S}\right| \overset{(3)}{\geq} |V_1| + \frac{\left|V_{1/2}\right|}{\Delta\left(G\left[V_{1/2}\right]\right)} \geq |V_1| + \frac{\left|V_{1/2}\right|}{\Delta(G)} \tag{4}$$

[8] On the given graph $G(V,E)$, a coloring of V consists of coloring the vertices of V in such a way that no adjacent vertices receive the same color; in other words, a color has to be an independent set and a coloring of V is, indeed, a partition of V into independent sets.

Finally, denote by $S^*_{1/2}$ a maximum independent set in $G[V_{1/2}]$. Observe that the value of the optimal solution for MAX INDEPENDENT SET-R in $G[V_{1/2}]$ is equal to $V_{1/2}/2$. Since MAX INDEPENDENT SET is a maximization problem, the objective value of (integer) MAX INDEPENDENT SET is bounded above by the objective value of (continuous) MAX INDEPENDENT SET-R. Hence:

$$\left|S^*_{1/2}\right| \leq \frac{\left|V_{1/2}\right|}{2} \tag{5}$$

Denote by $\alpha(G)$, the size of S^*. Then, obviously, the following holds:

$$opt(G) = \left|S^*\right| = \left|V_1\right| + \left|S^*_{1/2}\right| \overset{(5)}{\leq} \left|V_1\right| + \frac{\left|V_{1/2}\right|}{2} \tag{6}$$

Putting together (4) and (6) we get the claimed ratio and complete the proof of the theorem.

A slight improvement of the ratio claimed in Theorem 2 appears in [63].

An immediate corollary of Theorem 2 is that MAX INDEPENDENT SET belongs to Poly-APX. Also, since the value $\omega(G)$ of the worst solution to the problem is 0 (i.e., we can consider the empty vertex-set as a feasible MAX INDEPENDENT SET solution), standard- and differential-approximation ratios coincide. So, MAX INDEPENDENT SET belongs to Poly-DAPX also. Finally, let us note that the strongest approximation results known for this problem are the following:

- MAX INDEPENDENT SET is asymptotically approximable (i.e., for large $\Delta(G)$ within ratio $k/\Delta(G)$, for every fixed constant k ([28, 64]);
- MAX INDEPENDENT SET is approximable within ratio $O(\log^2 n/n)$ ([39]) and within $O(\log n/\Delta(G)\log\log n)$ ([29]);
- MAX INDEPENDENT SET is inapproximable within better than $O(n^{\varepsilon-1})$ for any $\varepsilon > 0$, unless P = NP ([42]).

4.2. Min set cover

Given a ground set $C = \{c_1,\ldots,c_n\}$ and a collection $S = \{S_1,\ldots,S_m\} \subset 2^C$ of subsets of C, MIN SET COVER consists of determining a minimum-size sub-collection $S' \subseteq S$ that covers C, i.e., such that $\cup_{S\in S'} S = C$.

Let us consider the following natural greedy algorithm for MIN SET COVER denoted by GREEDYSC:

1. set: $S' = S' \cup \{S\}$, where $S \in_{S_i \in S} \{|S_i|\}$ (S' is assumed to be empty at the beginning of the algorithm);
2. update $I(S,C)$ by setting: $S = S \setminus \{S\}$, $C = C \setminus S$ and, for $S_j \in S$, $S_j = S_j \setminus S$;
3. repeat steps 1 and 2 until $C = \varnothing$;
4. output S'.

This algorithm has been independently analyzed by [46, 52]. Its version for WEIGHTED MIN SET COVER where we ask for determining a set cover of minimum total weight has been analyzed by [20]. It is easy to see that Algorithm GREEDYSC runs in polynomial time.

Theorem 3. *The standard-approximation ratio of Algorithm GREEDYSC is bounded above by* $1+\ln\Delta$, *where* $\Delta=\max_{S\in\mathcal{S}}\{|S|\}$.

Proof. Denoted by $I_i(S_i,C_i)$, the surviving instance in the first moment residual cardinalities of sets in S are at most i; denoted by $m(I_i,S')$ the number of sets of residual cardinality i placed in S' and note that $m(I_\Delta,S')=m(I,S')$. Following these notations:

$$C_\Delta = C \tag{7}$$

$$m(I,S')=\sum_{i=1}^{\Delta} m(I_i,S') \tag{8}$$

For $i=1,\ldots,\Delta$, we have the following Δ-line equation-system:

$$|C_i| = \sum_{k=1}^{i} k\times m(I_k,S') \tag{9}$$

where any of the above equations expresses the facts that: (1) any time a set S is chosen to be part of S', Algorithm GREEDYSC removes from C the elements of S and (2) for any i, the remaining ground set C_i to be covered, is covered by sets of cardinality at most equal to i chosen later by the algorithm.

Multiplying the Δth line of (9) by $1/\Delta$ and, for the other lines, line i by $1/(i(i+1))$ and taking into account (7), (8) and the fact that:

$$\frac{1}{i(i+1)}=\frac{1}{i}-\frac{1}{i+1}$$

we finally obtain:

$$\left(\sum_{i=1}^{\Delta-1}\frac{|C_i|}{i(i+1)}\right)+\frac{|C|}{\Delta}=\sum_{i=1}^{\Delta} m(I_i,S')=m(I,S') \tag{10}$$

Consider now an optimal solution S^* of $I(S,C)$ and let S_i^* be an optimal solution for $I_i(S_i,C_i)$, $i=1,\ldots,\Delta$. Elements of S^* covering C_i are always present and form a feasible solution for I_i. Therefore:

$$opt(I_i) = |S_i^*| \leqslant |S^*| = opt(I) \tag{11}$$

$$|C_i| \leq i\times opt(I_i) \tag{12}$$

where (12) expresses the fact that in order to cover $|C_i|$ elements by sets covering at most i of them, at least $|C_i|/i$ such sets will be needed. Putting (10), (11) and (12) together, we get:

$$m(I,S) \leq opt(I) \times \sum_{i=1}^{\Delta} \frac{1}{i} \leq opt(I)(1+\ln \Delta)$$

that is the ratio claimed.

Let us now show that the ratio proved in Theorem 3 is asymptotically tight.

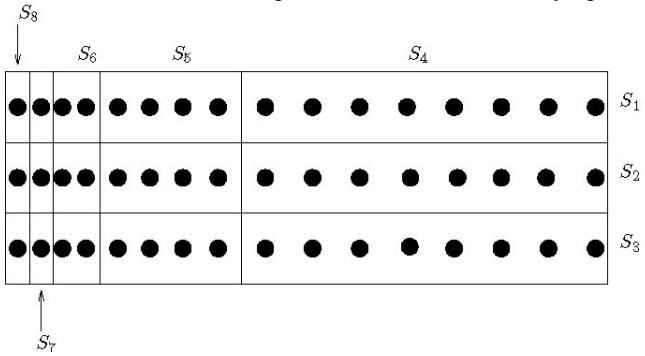

Figure 4: On the tightness of GREEDYSC with $k = 4$.

Some k is fixed and the instance of Figure 4, designed for $k = 4$, is considered. We are given a set C of 3×2^k elements: $C = \{1,2,\ldots,3\times 2^k\}$ and a family $S = \{S_1, S_2, \ldots, S_{k+4}\}$ of $k+4$ subsets of C.

For the reasons of simplicity, we shall consider that the elements of C are entries of a $3\times(k+1)$ matrix. Then, the sets in S are as follows:

- three disjoint sets: S_1, S_2 et S_3, each one containing the 2^k elements of each of the three lines of the matrix; they form a partition on C, i.e., $C = S_1 \cup S_2 \cup S_3$ et $S_i \cap S_j = \varnothing$, $i, j = 1,2,3$;
- $k+1$ sets $S_4,\ldots,S_{k+4}$, of respective sizes: $3\times 2^{k-1}, 3\times 2^{k-2}, \ldots, 3\times 2^1, 3\times 2^0$ and 3 contain, respectively, the points of the $2^{k-1}, 2^{k-2}, \ldots, 2^1, 2^0$ and 1 columns.

It is easy to see that Algorithm GREEDYSC will choose the $k+1$ sets $S_4,\ldots,S_{k+4}$, in this order, returning a set cover of size $k+1$, while the optimal set cover is the family $\{S_1, S_2, S_3\}$ of size 3. Note finally that, for the instance considered, $\Delta = 3\times 2^{k-1}$. Consequently, the approximation ratio achieved here is $(k+1)/3 = O(\ln(3\times 2^{k-1}))$.

More recently, another analysis of Algorithm GREEDYSC has been presented by [71] providing a tight ratio of $O(\log|C|)$. On the other hand, as stated in [67] (see [31] for an informal proof), it is impossible to approximate MIN SET COVER within better than $O(\log|C|) - O(\log\log|C|)$, unless a highly unlikely complexity classes

relationship is true. Note finally that the result of Theorem 3 has been slightly improved down to $\ln\Delta + (5/6)$ by [36].

A direct corollary of Theorem 3 (or alternatively of [71]) and of [67] is that MIN SET COVER definitely belongs to Log-APX.

Dealing with the differential approximation paradigm, MIN SET COVER is approximable within differential ratio bounded below by $1.365/\Delta$ ([12]) and inapproximable as is ([27]). So, it belongs to Poly-DAPX.

4.3. Min vertex cover

Given a graph $G(V,E)$, MIN VERTEX COVER consists of determining a minimum-size set $V' \subseteq V$ so that, for every $(u,v) \in E$, at least one among u and v belongs to V'.

Consider the following algorithm, denoted by MATCHING:

1. compute a maximal matching[9] M in G;
2. return the set C of the endpoints of the edges in M.

Algorithm MATCHING is polynomial since a maximal matching can be easily computed by picking an edge e, deleting it from G together with any edge sharing an endpoint with e and iterating these operations until no edge survives in G.

Theorem 4. *Algorithm MATCHING is a 2-standard-approximation algorithm for MIN VERTEX COVER.*

Proof. Let us first prove that C is a vertex cover for G. The endpoints of any edge e of M cover e itself and any other edge sharing a common endpoint with e. Denote by $V(M)$ the set of endpoints of the edges of M. Since M has been built to be maximal, the edges in M have common endpoints with any edge in $E \setminus M$, so, $V(M)$ cover both M and $E \setminus M$, i.e., the whole of E.

Set $m = |M|$; then:

$$|V(M)| = m(G,C) = 2m \tag{13}$$

On the other hand, since edges in M do not share common endpoints pairwise, any solution (a fortiori an optimal one) must use at least m vertices in order to cover them (one vertex per edge of M). So, denoting by $\tau(G)$ the cardinality of a minimum vertex cover in G, we have:

$$\tau(G) = opt(G) \geq m \tag{14}$$

Putting (13) and (14) together, we immediately get: $m(G,C)/\tau(G) \leq 2$ as claimed.

---

[9] Given a graph $G(V,E)$, a matching is a subset $M \subseteq E$ such that no two edges in M share a common endpoint; the matching is maximal (for inclusion) if it cannot be augmented when remaining a matching.

We now show that the ratio claimed in Theorem 4 is tight. Let us consider the graph of Figure 5. The thick edge is the maximal matching for this graph and, consequently, the solution taken by the Algorithm MATCHING will contain two vertices (the two endpoints of the thick edge). On the other hand, the center of the star suffices to cover all its edges.

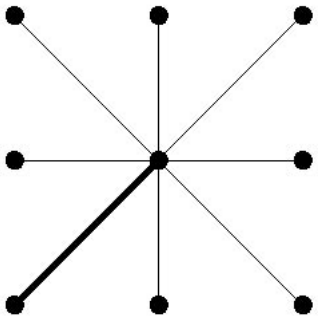

Figure 5: Tightness of the ratio achieved by Algorithm MATCHING.

One of the best known open problems in polynomial approximation is the improvement of ratio 2 for MIN VERTEX COVER. A lot of unsuccessful effort has been put into such improvement until now. All this long effort has only produced ratios of the form:

- $2-(\log\log n/\log n)$ ([10, 54]);
- $2-(2\ln\ln n/\ln n)$ ([40]);
- $2-(\log\log\Delta(G)/\log\Delta(G))$ ([40]).

Unfortunately, a more recent result by [50] gives strong evidence that this improvement is likely to be impossible.

In the differential approximation paradigm, MIN VERTEX COVER is equiapproximable with MAX INDEPENDENT SET ([27]). So, it definitely belongs to Poly-DAPX.

4.4. Min TSP

Given a complete graph on n vertices, denoted by K_n, with positive weights on its edges, MIN TSP consists of determining a Hamiltonian tour[10] of K_n of minimum total cost.

Let us note that, with respect to the differential paradigm, computing the worst solution for MIN TSP is not trivial at all. As opposed to the problems seen in the previous sections that had "trivial" worst solutions (the empty set for MAX INDEPENDENT SET, the whole family S for MIN SET COVER, or the whole vertex-set V of the input-graph for

[10] A simple cycle passing through all the vertices of K_n.

MIN VERTEX COVER) the worst solution of MIN TSP is an optimal solution of MAX TSP, where we wish to determine a maximum total-cost Hamiltonian cycle. This problem is also NP-hard. So, determining the worst solution of MIN TSP is as hard as to determine an optimal solution.

Let us consider the following well-known algorithm for MIN TSP, denoted by 2_OPT, originally devised by [25], where $d(i,j)$ denotes the weight of edge (v_i, v_j):

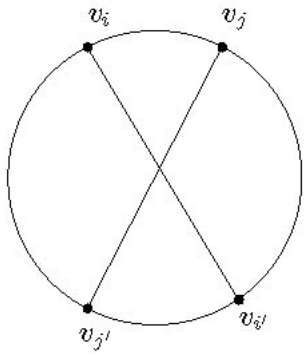

Figure 6: Algorithm 2_OPT

1. construct some Hamiltonian tour T (this can be done, for example, by the nearest-neighbor heuristic);
2. consider the two edges (v_i, v_j) and $(v_{i'}, v_{j'})$ of T; if $d(i,j)+d(i',j') > d(i,i')+d(j,j')$, then replace (v_i, v_j) and $(v_{i'}, v_{j'})$ in T by $(v_i, v_{i'})$ and $(v_j, v_{j'})$ (Figure 6) i.e., produce a new Hamiltonian tour $T \setminus \{(v_i, v_j), (v_{i'}, v_{j'})\} \cup \{(v_i, v_{i'}), (v_j, v_{j'})\}$;
3. repeat step 2 until no swap is possible;
4. output T the finally produced tour.

Algorithm 2_OPT is polynomial when, for instance, $d_{\max}$ the maximum edge-weight is bounded above by a polynomial of n. For other cases where 2_OPT is polynomial, see [56].

Theorem 5. *([56]) The algorithm 2_OPT achieves a differential-approximation ratio bounded below by 1/2.*

Proof. Assume that T is represented as the set of its edges, i.e.:

$$T = \{(v_1, v_2), \ldots, (v_i, v_{i+1}), \ldots, (v_n, v_1)\}$$

and denote by T^* an optimal tour. Let $s^*(i)$ be the index of the successor of v_i in T^*. So, $s^*(i)+1$ is the index of the successor of $v_{s^*(i)}$ in $T \pmod n$ (in other words, if $s^*(i) = j$, then $s^*(i)+1 = j+1$).

The tour T computed by 2_OPT is a local optimum for the 2-exchange of edges in the sense that every interchange between the two non-intersecting edges of T and the

two non-intersecting edges of $E \setminus T$ will produce a tour of total distance at least equal to $d(T)$, where $d(T)$ denotes the total weight of T. This implies in particular that, $\forall i \in \{1,\ldots,n\}$:

$$d(i,i+1)+d\left(s^*(i),s^*(i)+1\right) \leq d\left(i,s^*(i)\right)+d\left(i+1,s^*(i)+1\right) \tag{15}$$

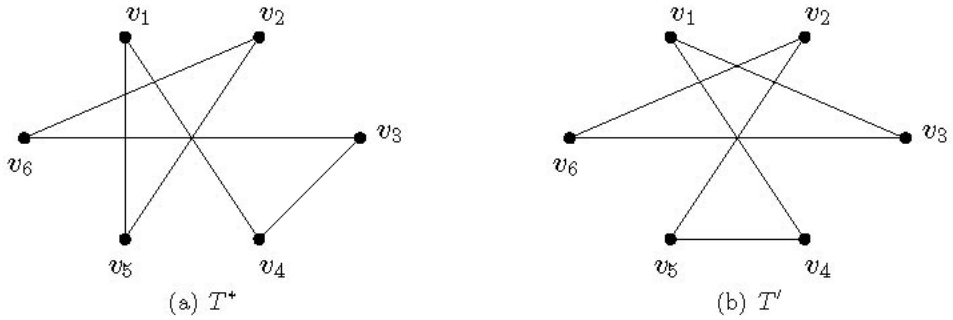

Figure 7: Tours T^* and T' of the proof of Theorem 5 for a K_6.

Observe now that (see also Figure 7), denoted by T_w the worst tour in K_n the following holds:

$$\bigcup_{i=1,\ldots,n} \{(v_i,v_{i+1})\} = \bigcup_{i=1,\ldots,n} \left\{\left(v_{s^*(i)},v_{s^*(i)+1}\right)\right\} = T \tag{16}$$

$$\bigcup_{i=1,\ldots,n} \left\{\left(v_i,v_{s^*(i)}\right)\right\} = T^* \tag{17}$$

$$\bigcup_{i=1,\ldots,n} \left\{\left(v_{i+1},v_{s^*(i)+1}\right)\right\} = \text{some feasible tour } T' \text{ better than } T_\omega \tag{18}$$

Add inequalities in (15), for $i = 1,\ldots,n$:

$$\sum_{i=1}^{n}\left(d(i,i+1)+d\left(s^*(i),s^*(i)+1\right)\right) \leq \sum_{i=1}^{n}\left(d\left(i,s^*(i)\right)+d\left(i+1,s^*(i)+1\right)\right) \tag{19}$$

Putting (19) together with (16), (17) and (18) we get:

$$(16) \Rightarrow \sum_{i=1}^{n} d(i,i+1)+\sum_{i=1}^{n} d\left[s^*(i),s^*(i)+1\right] = 2m\left(K_n,T\right)$$

$$(17) \Rightarrow \sum_{i=1}^{n} d\left[i,s^*(i)\right] = opt\left(K_n\right)$$

$$(18) \Rightarrow \sum_{i=1}^{n} d\left[i+1, s^*(i)+1\right] = d\left(T'\right) \leq \omega(K_n)$$

Then, some easy algebra, and taking into account that the differential ratio for a minimization problem increases with ω, lead to the differential ratio claimed and completes the proof of the theorem.

We shall now show that the ratio 1/2 is tight for the Algorithm 2_OPT. Let us consider a K_{2n+8}, $n \geq 0$, set $V = \{i : i = 1, \ldots, 2n+8\}$, set:

$$d(2k+1, 2k+2) = 1 \quad k = 0, 1, \ldots, n+3$$
$$d(2k+1, 2k+4) = 1 \quad k = 0, 1, \ldots, n+2$$
$$d(2n+7, 2) = 1$$

and set the distances of all the remaining edges to 2.

Consider the tour $T = \{(i,(i+1)) : i = 1, \ldots, 2n+7\} \cup \{((2n+8),1)\}$ and observe that it is a local optimum for the 2-exchange on K_{2n+8}. Indeed, let $(i,(i+1))$ and $(j,(j+1))$ be the two edges of T. We can assume w.l.o.g. that $2 = d(i,i+1) \geq d(j,j+1)$, otherwise, the cost of T cannot be improved. Therefore, $i = 2k$, for some k.

In fact, in order for the cost of T to be improved, there are two possible configurations, namely $d(j,j+1) = 2$ and $d(i,j) = d(j,j+1) = d(i+1,j+1) = 1$. Thus, the following assertions hold:

- if $d(j,j+1) = 2$, then $j = 2k'$, for some k', and, according to the construction of K_{2n+8}, $d(i,j) = 2$ (since i and j are even), and $d(i+1,j+1) = 2$ (since $i+1$ and $j+1$ are odd); so the 2-exchange does not yield a better solution;
- if $d(i,j) = d(j,j+1) = d(i+1,j+1) = 1$, then according to the construction of K_{2n+8} we will have $j = 2k'+1$ and $k' = k+1$; so, we lead a contradiction since $1 = d(i+1,j+1) = 2$.

Furthermore, one can easily see that the tour:

$$T^* = \{(2k+1)(2k+2) : k = 0, \ldots, n+3\} \cup \{(2k+1)(2k+4) : k = 0, \ldots, n+2\}$$
$$\{(2n+7)2\}$$

is an optimal tour of value $opt(K_{2n+8}) = 2n+8$ (all its edges have distance 1) and that the tour:

$$T_\omega = \{(2k+2)(2k+3) : k = 0, \ldots, n+2\} \cup \{(2k+2)(2k+5) : k = 0, \ldots, n+1\}$$
$$\cup \{(2n+8)1, (2n+6)1, (2n+8)3\}$$

realizes a worst solution for K_{2n+8} with value $\omega(K_{2n+8}) = 4n+16$ (all its edges have distance 2).

Consider a K_{12} constructed as described just above (for $n=2$). Here, $d(1,2)=d(3,4)=d(5,6)=d(7,8)=d(9,10)=d(11,12)=d(1,4)=d(6,3)$ $=d(5,8)=d(7,10)=d(9,12)=d(11,2)=1$

while all the other edges are of distance 2. In Figures 8(a) and 8(b), T^* and T_ω, respectively, ($T=\{1,\ldots,11,12,1\}$) are shown. Hence, in K_{2n+8} considered, the differential-approximation ratio of 2_OPT is equal to 1/2.

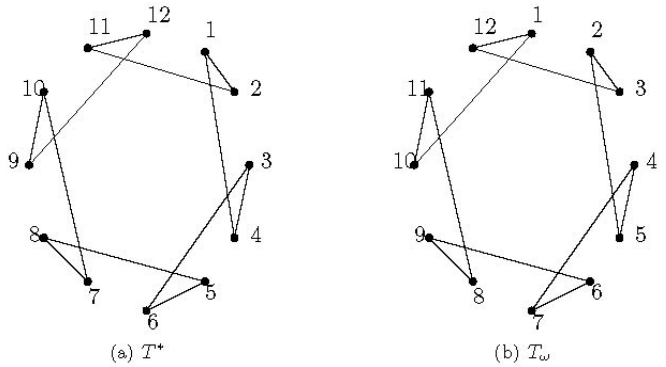

Figure 8: Tightness of the 2_OPT approximation ratio for $n=1$.

The best differential-approximation ratio for MIN TSP is 3/4 ([30]) but it does not admit a polynomial time differential-approximation schema ([56, 55]. So, MIN TSP definitely belongs to DAPX. Furthermore, it is proved in [56] that MIN TSP and MIN METRIC TSP are equiapproximable for the differential approximation.

Dealing with the standard approximation paradigm, the MIN TSP is in Exp-APX (see Section 6). On the other hand, the MIN METRIC TSP is approximable within standard-approximation ratio 3/2 by the celebrated Christofides algorithm ([19]) while the most famous relaxation of MIN METRIC TSP, that is when the edge-weights are either 1 or 2 is approximable within 8/7 ([17]). Finally, the MIN EUCLIDEAN TSP is in PTAS ([1]).

4.5. Min coloring

Given a graph $G(V,E)$, MIN COLORING consists of determining the minimum number of colors (i.e., of independent sets, see also footnote 8), that feasibly color the vertices of G.

The worst-solution value for MIN COLORING is equal to n, since coloring any vertex of the input graph with its own color produces a feasible coloring. Furthermore, this coloring cannot be augmented without producing empty colors.

Consider the following algorithm for MIN COLORING, denoted by COLOR and devised by [41] (see also [65]):

1. find an independent set S of size 3 in G; color its vertices with a new color and remove it from G;
2. repeat step 1 until no independent set of size 3 is found;
3. determine a maximum family of disjoint independent sets of size 2 in (the surviving) graph G and color the vertices of each of them with a new color;
4. color the remaining vertices of G using as new colors as the vertices to be colored;
5. output C the union of colors used at steps 2, 3 and 4.

Observe first that Algorithm COLOR runs in polynomial time. Indeed, step 1 is greedy. For a graph of order[11] n, all the independent sets of size 3 can be found in time $O(n^3)$ by an exhaustive search. Step 2 can be performed in polynomial time since it amounts to a maximum matching computation that is polynomial ([61]). Indeed, at step 3, the maximum independent set of the surviving graph G has size at most 2. Consider $\overline{G}$ that is the complementary[12] of G. Any independent set of size 2 in G becomes an edge in $\overline{G}$ and maximum family of disjoint independent sets of size 2 in G is exactly a maximum matching in $\overline{G}$. So, computation in step 3 is nothing else than computation of a maximum matching.

Lemma 1. *Steps 3 and 4 of Algorithm COLOR optimally color a graph G with $\alpha(G) = 2$.*

Proof. Since $\alpha(G) = 2$, colors in G are either independent sets of size 2, or singletons. Fix some coloring C using x colors of size 2 and y colors that are single vertices. If n is the order of G, we have:

$$|C| = x + y \tag{20}$$

$$n = 2x + y \tag{21}$$

By (20) and (21), $|C| = n - x$. Hence, the greater the x, the better the coloring C and a minimum coloring corresponds to a maximum x. This is exactly what Step 3 of Algorithm COLOR does.

We are ready now to prove the following theorem.

Theorem 6. ([41]) *The algorithm COLOR is a 2/3-differential-approximation algorithm for MIN COLORING.*

Proof. We prove the theorem by induction on *n*, the size of the input graph.

If *n = 1,* then Algorithm COLOR optimally colors it with one color. Assume that theorem's statement remains true for $n \leq k$ and consider a graph G of order $n = k + 1$. We distinguish two cases.

---

[11] The order of a graph is the cardinality $|V|$ of its vertices.

[12] Given a graph $G(V, E)$, the complementary graph $\overline{G}(V, \overline{E})$ of G is the graph having the same set of vertices V as G and $\overline{E} = \{(v_i, v_j) : i \neq j \text{ and } (v_i, v_j) \notin E\}$.

If G does not contain an independent set of size greater than 2, then by Lemma 1, Algorithm COLOR computes an optimal coloring for G.

Assume now that G is such that $\alpha(G) \geq 3$ and denote by $\chi(G)$ the chromatic number (i.e., the cardinality of a minimum coloring) of G. Then, obviously, at least an independent set S has been found by COLOR at step 1 and:

$$\chi(G[V \setminus S]) \leq \chi(G) \tag{22}$$

Consider the graph $G[V \setminus S]$ of order $n-3$ and its coloring $C \setminus S$. This graph is colored with $|C|-1$ colors and, by the induction hypothesis:

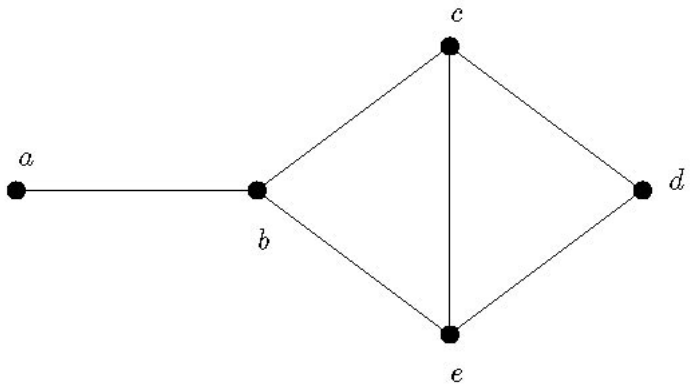

Figure 9: The complement of a graph where differential ratio 2/3 is attained by Algorithm COLOR.

$$n-3-|C \setminus S| \geq \frac{2}{3}(n-3-\chi(G[V \setminus S])) \tag{23}$$

Combining (22) and (23) we get:

$$n-|C| = n-|C \setminus S|-1 \geq \frac{2}{3}(n-3-\chi(G[V \setminus S]))+2 \geq \frac{2}{3}(n-\chi(G)) \tag{24}$$

Taking into account that $\omega(G) = n$, (24) directly derives the differential ratio claimed.

We now prove that the differential ratio 2/3 is tight for the Algorithm COLOR. Consider a graph G the complement of which is shown in Figure 9. It is easy to see that in G, Algorithm COLOR would produce $C = \{\{a\},\{d\},\{b,c,e\}\}$, while the optimal coloring is $C^* = \{\{a,b\},\{c,d,e\}\}$. Taking into account that $\omega(G) = 5$, ratio 2/3 for this instance is immediately proved.

Dealing with standard paradigm, the best known approximation ratios for MIN COLORING are:

- $O(n(\log\log n)^2/\log^3 n)$ ([38]);
- $\Delta \log\log n/\log\Delta$ ([29]).

On the other hand, it is inapproximable within better than:

- $n^{1-\varepsilon}$ for any constant $\varepsilon > 0$, unless $NP \subseteq coRP$ ([34]);
- $n^{(1/5)-\varepsilon}$ for any $\varepsilon > 0$, assuming that $NP \neq coRP$ ([15]);
- $n^{(1/7)-\varepsilon}$ for any $\varepsilon > 0$ unless $P \neq NP$ ([15]).

More about the complexity classes mentioned in the results above can be found in [59].

4.6. Min weighted bipartite coloring

Consider a vertex-weighted graph $G(V,E)$ and denote by w_i the weight of vertex $v_i \in V$. For any subset $V' \subseteq V$ define its weight $w(V')$ by:

$$w(V') = \max\{w_i : v_i \in V'\} \tag{25}$$

MIN WEIGHTED COLORING consists of determining a coloring $C = (S_1,\ldots,S_k)$ of G minimizing the quantity:

$$m(G,C) = \sum_{i=1}^{k} w(S_i) \tag{26}$$

where, for $i = 1,\ldots,k$, $w(S_i)$ is defined as in (25).

MIN WEIGHTED COLORING is obviously NP-hard in general graphs since setting $w_i = 1$, $v_i \in V$, it becomes the classical MIN COLORING problem. However, it is proved in [26] that it is NP-hard, even in bipartite graphs (MIN WEIGHTED BIPARTITE COLORING). Let us note that MIN COLORING is polynomial in these graphs since they are 2-colorable.

In what follows in this section, we shall present a polynomial time differential-approximation scheme originally developed by [26].

Consider a vertex-weighted bipartite graph $B(U,D,E)$ and the following algorithm denoted by BIC:

1. range the vertices of B in decreasing order with respect to their weights;
2. fix an $\varepsilon > 0$ and set $\eta = \lceil 1/\varepsilon \rceil$; set $S_U = \{v_{4\eta+3},\ldots,v_n\} \cap U$ and $S_D = \{v_{4\eta+3},\ldots,v_n\} \cap D$;
3. compute an optimal weighted coloring $\tilde{C}$ in $B' = B[\{v_1,\ldots,v_{4\eta+2}\}]$;
4. output $C = S_U \cup S_D \cup \tilde{C}$.

Since the graph B' of step 3 has a fixed size, the computation of $\tilde{C}$ can be performed in polynomial time by an exhaustive search. So, Algorithm BIC is polynomial.

Denote by $C^* = (S_1^*, S_2^*,\ldots,S_p^*)$ an optimal MIN WEIGHTED BIPARTITE COLORING-solution of B and let $w_1 = w_{i_1} \geq w_{i_2} \geq \ldots \geq w_{i_p}$ be the weights of its colors. Remark also that:

$$\begin{aligned} \omega(B) &= \sum_{v_i \in U \cup D} w(v_i) \\ opt(B) &= w_{i_1} + w_{i_2} + \ldots + w_{i_p} \\ \omega(B') &= \sum_{i=1}^{4\eta+2} w_i \qquad (27) \\ \omega(B') &\leqslant \omega(B) \qquad (28) \end{aligned}$$

The proof of the existence of a polynomial time differential-approximation scheme for MIN WEIGHTED BIPARTITE COLORING is based upon the following two lemmata.

Lemma 2. $|\tilde{C}| \leq 2\eta + 2$.

Proof. Note first that it cannot exist more than 2 colors that are singletons in $\tilde{C}$. A contrario, at least two of them are in U or in D. By concatenating them into a single color we reduce the objective value (26) of $\tilde{C}$.

Denote by x the number of colors that are singletons and by y the number of the other colors of $\tilde{C}$. Then, obviously, $x + 2y \leq 4\eta + 2$ and, as mentioned just before, $x \leq 2$; henceforth:

$$2x + 2y \leq 4\eta + 4 \Rightarrow \left|\tilde{C}\right| = x + y \leq 2\eta + 2 \text{ that proves the lemma.}$$

Lemma 3. $m(B', \tilde{C}') = opt(B') \leqslant opt(B)$.

Proof. Just remark that coloring $(S_1^* \cap V(B'), S_2^* \cap V(B'), \ldots, S_p^* \cap V(B'))$ is feasible for B' and it is only a part of C^*.

We are ready now to prove the following theorem.

Theorem 7. *([26]) The algorithm BIC is a polynomial time differential-approximation schema for the MIN WEIGHTED BIPARTITE COLORING.*

Proof. Using (27) and Lemma 2 we have:

$$\omega(B') - opt(B') = \sum_{i=1}^{4\eta+2} w_i - \sum_{j=1}^{|\tilde{C}|} w_{i_j} \geqslant 2\eta w_{4\eta+2} \geqslant \frac{2}{\epsilon} w_{4\eta+2} \qquad (29)$$

On the other hand:

$$w(S_U) \leq w_{4\eta+2} \qquad (30)$$

$$w(S_D) \leq w_{4\eta+2} \qquad (31)$$

From (29), (30) and (31), we get:

$$\begin{aligned} m(B,C) &= w(S_U) + w(S_D) + opt(B') \\ &= (1-\epsilon)opt(B') + \epsilon\left(opt(B') + \frac{1}{\epsilon}w(S_U) + \frac{1}{\epsilon}w(S_D)\right) \\ &\leqslant (1-\epsilon)opt(B') + \epsilon\left(\underbrace{opt(B') + \frac{2}{\epsilon}w_{4\eta+2}}_{\leqslant\omega(B')\leqslant\omega(B)}\right) \leqslant (1-\epsilon)opt(B) + \epsilon\omega(B) \end{aligned}$$

that gives the schema claimed.

As shown in [26], the MIN WEIGHTED BIPARTITE COLORING cannot be solved by fully polynomial time differential-approximation scheme. On the other hand, it is approximable within standard-ratio slightly better that 4/3 and inapproximable within standard ratio 8/7, unless **P** = *NP* ([26]).

4.7. Knapsack

We finish Section 4 by handling one of the most famous problems in combinatorial optimization, that is KNAPSACK. An instance I of KNAPSACK is the specification of two vectors $\vec{a}$ and $\vec{c}$ and of a constant b and can be defined in terms of an integer-linear program as follows:

$$I = \begin{cases} \max & \vec{a}\cdot\vec{x} \\ & \vec{c}\cdot\vec{x} \leq b \end{cases}$$

Consider the following algorithm for KNAPSACK presented by [45]:

1. fix an $\varepsilon > 0$ and build the instance $I' = ((a_i', c_i)_{i=1,\ldots,n}, b)$ with $a_i' = \lfloor a_i n/(a_{\max}\varepsilon)\rfloor$;
2. output $S :=$ DYNAMICPROGRAMMING(I′).

This dynamic programming algorithm is a classical example of how polynomial time approximation schemata are constructed. In fact, the most common technique for them consists first of scaling down data in such a way that the new instance becomes polynomial, then of solving it and, finally, of proving that the solution obtained corresponds to a feasible solution of the initial instance whose value is "very close" to the optimal value.

Step 2 above runs in $O(n^2 a'_{\max}\log c_{\max}) = O((n^3\log c_{\max})/\varepsilon)$ ([45]). So the whole running time of the algorithm is polynomial.

Theorem 8. *KNAPSACK* $\in$ *FPTAS.*

Proof. Let S^* be an optimal solution of I. Obviously, S^* is feasible for I'. Let:

$$t = \frac{a_{\max}\varepsilon}{n} \tag{32}$$

Then, for every $i = 1,\ldots,n$:

$$a_i' = \left\lfloor \frac{a_i}{t} \right\rfloor \tag{33}$$

and the following holds:

$$opt(I') \geq \sum_{i \in S^*} a'_i \geq \sum_{i \in S^*} \left(\frac{a_i}{t} - 1 \right)$$

$$\geq \frac{opt(I)}{t} - \left| S^* \right| \geq \frac{opt(I)}{t} - n \Rightarrow t\, opt(I') \geq opt(I) - nt \tag{34}$$

Note now that the largest a_i, i.e., that whose index verify: $i_0 = \text{argmax}\{a_{\max}\}$ is feasible for I. Hence,

$$opt(I) = \sum_{i \in S^*} a_i \geq a_{\max} \tag{35}$$

Putting (32), (35) and (34) together, we get:

$$nt = a_{\max} \varepsilon \leq \varepsilon(I) \tag{36}$$

Then the following hold for the value of the solution S returned by the algorithm:

$$m(I,S) = \sum_{i \in S} a_i \overset{(33)}{\geq} t \sum_{i \in S} a_{i'} = t\, opt(I') \overset{(34)}{\geq} opt(I) - nt \overset{(36)}{\geq} (1-\varepsilon) opt(I)$$

To conclude, it suffices to observe that the complexity of the algorithm is "fully" polynomial, since it does not depend on ε. Moreover, since it depends on the logarithm of $c_{\max}$, the algorithm remains polynomial even if $c_{\max}$ is exponential with the size n of the instance. The proof of the theorem is completed.

Let us note that, taking that nothing is feasible for KNAPSACK, producing a solution of value 0 that is the worst solution for any instance I. So, standard and differential approximation ratios coincide for KNAPSACK. Henceforth, KNAPSACK belongs also to DFPTAS.

5. APPROXIMABILITY PRESERVING REDUCTIONS

The transformation of a problem into a different, but related, problem with the aim of exploiting the information we have on the latter in order to solve the former, has always been present in mathematics. Consider, for example, how Greek mathematicians and, subsequently, Al Khuwarizmi ([14]) made use of geometrical arguments in order to solve algebraic problems.

In recent times, a particular type of transformation, called *reduction* has been introduced by logicians in computability theory ([51]). In this case, a reduction from a problem Π to a problem Π' not only specifies how the former can be solved starting from the solution of the latter but, possibly more important in such context, it allows showing that if problem Π is unsolvable (i.e., no algorithm for its solution may exist), and so is problem Π'. Such a development of the notion of problem transformation is of great importance because it determines a twofold application of mathematical transformations: on one side they allow transfering positive results (solution techniques)

from one problem to another and on the other side they may also be used for deriving negative (impossible) results.

The first application of the concept of reduction in computer science and combinatorics arose in the early seventies in the seminal paper by [21], and was soon followed by the equally fundamental paper by [47]. Actually, both Cook's and Karp's reductions where conceived to relate to decision problems from a complex, theoretical point of view. So, if we want to use reductions in solving optimization problems, we need other types of more "optimization-oriented" reductions.

Why do we need them? NPO hierarchy discussed in Section 2 (Figures 1 and 2) has been built in a somewhat ad-hoc and "absolute" way in the sense that problems are (constructively) classified following algorithms solving them. However, this classification does not allow comparisons between approximability properties of problems. For instance, we cannot answer or we can only partially answer the questions such as:

- how can one compare problems with respect to their approximability properties and independently on their respective approximation levels?
- how can one compare the approximability of different versions of the same problem (for example, weighted version vs. unweighted one)?
- how can one link different types of approximation for a same problem (for instance, do there exist transfers of approximability results between standard and differential approximation for a given problem)?
- how to apprehend the role of parameters in the study of approximability (for example, we have seen in Section 4.1 that the functions describing approximation ratios for MAX INDEPENDENT SET are different when dealing with n or when dealing with $\Delta(G)$)?
- can we transfer approximation results from one problem to another one?
- can we refine the structure of the approximation classes given above by showing, for instance, that some of the problems are harder than the other ones within the same approximability class (completeness of results)?

Researchers try to provide answers to these questions by using carefully defined reductions called the *approximation preserving reductions*. Any of the existing ones imposes particular conditions on the way to the optimal solutions, or the approximation ratios etc. are transformed from one problem to another. For more details, the interested reader can referred to [5, 8, 9, 22, 32, 44, 66, 72].

In general, given the two NPO problems Π = (***I***, Sol, m, goal) and Π' = (***I***, Sol, m, goal′), an approximation preserving reduction R from Π to Π' (denoted $\Pi \leq_R \Pi'$) is a triple (f, g, c) of polynomially computable functions such that:

- f transforms an instance $I \in \boldsymbol{I}$ into an instance $f(I) \in \boldsymbol{I}$;
- g transforms a solution $S' \in \text{Sol}'(f(I))$ into a solution $g(I, S') \in \text{Sol}(I)$;
- c transforms ratio $\rho'(f(I), S')$ into $\rho(I, g(I, S')) = c(\rho'(f(I), S'))$.

A basic property of an approximation preserving reduction $\Pi \leq_R \Pi'$ is that:

- if Π' is approximable within ratio ρ', Π is approximable within ratio $\rho = c(\rho')$;

- on the other hand, if, under a likely complexity hypothesis, Π is not approximable within ratio ρ, then (provided that c is invertible) Π' is not approximable within ratio $\rho' = c^{-1}(\rho)$.

Every reduction can be seen as a binary hardness-relation among problems.

The study and the use of approximation preserving reductions is interesting and relevant for both the theoretical computer science and the operational research communities for two main reasons.

The first reason is "structural". By means of these reductions, one refines the class of NP-hard problems by drawing a hierarchy of classes in the interior of NP-hard. This hierarchy can be seen as a sequence of strata, each stratum containing problems of "comparable approximability difficulty (or easiness)". Indeed, any stratum C draws the capacity of its problems to be approximable within the approximation level represented by C and, simultaneously, the limits to the approximability of these problems. For instance let us refer to Figure 10. Let us assume that the two strata C' and C" represent the two approximation levels and suppose, w.l.o.g. that C' is the class APX and C" is the class Log-APX. Let us also suppose that a new problem Π, for which no approximation result was known, has been reduced to a problem $\Pi' \in$ Log-APX by a reduction preserving logarithmic approximation ratios. An immediate corollary is then that Π also belongs to Log-APX and, unless a stronger positive result is proved for it, does not belongs to APX.

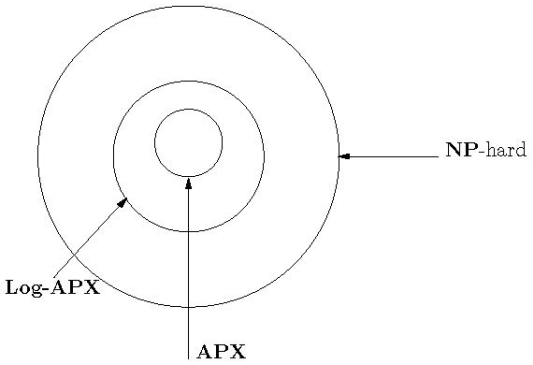

Figure 10: Designing an approximability hierarchy for NP-hard class.

The second reason is "operational". Approximability preserving reductions represent a kind of alternative in the achievement of new approximation results for particular hard problems. When one tries to approximately solve a new problem Π (or to improve existing approximation results for it) a possible way is to operate autonomously by undertaking a thorough and "from-the-beginning" study of the approximability of Π. However, another way to apprehend Π is to put in contribution not only the structural characteristics of this problem but also the whole knowledge dealing with all the problems "similar" to Π. For instance, assume that there exist two approximation preserving reductions R from Π to a problem Π' and Q from a problem Π'' to Π and that we know a positive approximation result for Π' (i.e., an algorithm achieving

approximation ratio r' for it), and a negative result for Π'' i.e, an assertion that Π'' is not approximable within some ratio r'', unless an unlikely complexity-theoretical assumption (for example, P = NP) holds. Then, according to the particular characteristics of R and Q, one derives that Π is approximable within ratio, say $c'(r')$, but it is not approximable within ratio $c''(r'')$, where c' and c'' are positive real functions depending on R and Q, respectively (see the definition of an approximability preserving reduction given above).

There exist a lot of approximability preserving reductions devised today for several combinatorial optimization problems. Let us give two very simple examples:

Example 1. MAX INDEPENDENT SET and MAX CLIQUE[13]. It is very well-known from graph theory that an independent set in a graph G becomes a clique of the same size in $\bar{G}$ and vice-versa.

Assume now that we know an approximation algorithm A for MAX INDEPENDENT SET achieving an approximation ratio expressed as function of n (the size of the input-graph). Assume also that we want to approximately solve MAX CLIQUE in a graph G. Then, we can build $\bar{G}$ and run A on it. Algorithm A will return an independent set of $\bar{G}$ that becomes a clique in G. This clique has the same size as the independent set initially computed and since G and $\bar{G}$ have the same size, the approximation ratio achieved for MAX INDEPENDENT SET is also achieved for MAX CLIQUE. The inverse is also true.

Example 2. MIN COLORING and MIN PARTITION INTO CLIQUES[14]. Using the relation between a clique and an independent set in Example 10, it can easily be seen that a coloring in G becomes a partition into cliques (of the same size) in $\bar{G}$. So, an approximation algorithm A for say MIN COLORING achieving an approximation ratio expressed as function of n can be used in the same way as previously to solve MIN PARTITION INTO CLIQUES (and vice-versa) with the same approximation ratio in both standard and differential paradigms.

For a long time, approximability-preserving reductions have been considered as a kind of "universal" tools allowing us to produce any kind of results and for any kind of approximation ratios (i.e., independently on their forms and parameters). But this is absolutely not true. In fact, reductions are not universal. Most of them cannot preserve neither every value nor every form of approximation ratio.

Let us revisit the reduction of Example 10. If, as we did there, we assume that the ratio is function of n, then preservation works. The same would hold if the ratio assumed was a constant. If, on the other hand, this ratio is a function of, say, $\Delta(G)$, then things become complicated, since no general relation exists between $\Delta(G)$ and $\Delta(\bar{G})$. So, reduction of Example 10 does not preserve ratios functions of the maximum degree of the input graph. The same observation can be also made for Example 10.

---

[13] Given a graph $G(V,E)$, MAX CLIQUE consists of determining a maximum-size subset $V' \subseteq V$ such that $G[V']$ is a complete graph.

[14] Given a graph G, MIN PARTITION INTO CLIQUES consists of determining a minimum partition of the vertex-set G into sets each of them inducing a complete subgraph of G, i.e., a clique.

Let us finally note that most of the reductions known are devised to preserve approximation ratios that are "better than constants".

As a last example, let us consider the following classical reduction between MAX WEIGHTED INDEPENDENT SET[15] and MAX INDEPENDENT SET ([70]).

Let us consider an instance $(G(V,E),\vec{w})$ of MAX WEIGHTED INDEPENDENT SET and suppose, in order that the reduction that follows is polynomial, that weights are polynomial with the order n of G. We transform it into an instance $G'(V',E')$ of MAX INDEPENDENT SET as follows:

- we replace every vertex $v_i \in V$ by an independent set W_i of w_i new vertices;
- we replace every edge $(v_i,v_j) \in E$ by a complete bipartite graph among the vertices of the independent sets W_i et W_j in G' (see Figure 11 where the vertices v_i and v_j have respectively weights 3 and 2).

This transformation is polynomial since the resulting graph G' has $\sum_{i=1}^{n} w_i$ vertices and every w_i is polynomial with n.

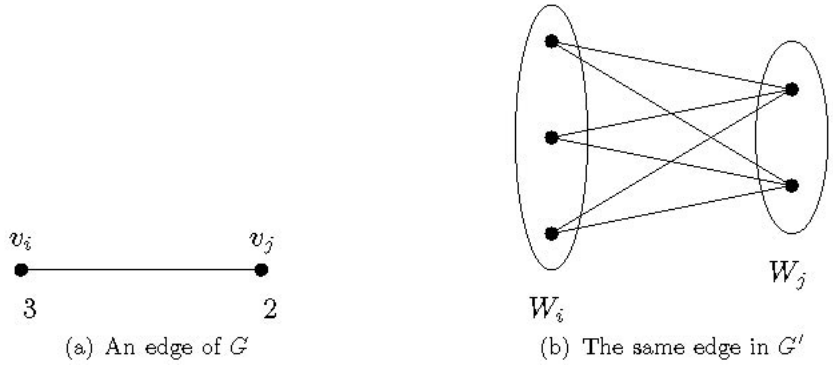

Figure 11: Transformation of an instance $(G(V,E),\vec{w})$ of MAX WEIGHTED INDEPENDENT SET into an instance $G'(V',E')$ of MAX INDEPENDENT SET.

Let us now consider an independent set S' of G' and w.l.o.g. let us assume it is maximal with respect to inclusion (in case it is not, we can easily add vertices until we reach a maximal independent set). Then, $S' = \cup_{j=1}^{k} W_{i_j}$, i.e., there exists a k such that S' consists of k independent sets W_{i_j}, $j = 1,\ldots,k$, corresponding to k independent vertices $v_{i_1},\ldots,v_{i_k} \in V$. So, $|S'| = \sum_{j=1}^{k} w_{i_j}$.

---

[15] Given a vertex-weighted graph G, the objective is to determine an independent set maximizing the sum of the weights of its vertices.

Hence, consider an independent set S' of G'. If W_i, $i=1,\ldots,k$, are the independent sets that form S' ($|S'|=\sum_{i=1}^{k} w_i$), then an independent set S for G can be built that contains all corresponding vertices $v_1,\ldots,v_k$ of V with weights $w_1,\ldots,w_k$, respectively. The total weight of S is then $\sum_{i=1}^{k} w_i = |S'|$

Let us now suppose that we have a polynomial time approximation algorithm A with ratio r for MAX INDEPENDENT SET and consider an instance of $(G(V,E),\vec{w})$ of MAX WEIGHTED INDEPENDENT SET. Then the following algorithm is a polynomial time approximation algorithm for MAX WEIGHTED INDEPENDENT SET, denoted by WA:

- construct G' as previously;
- run A on G'; let S' be the computed independent set;
- construct an independent set S of G as explained before.

From what has been discussed just above from any solution for S' in G' we can build a solution S for G of value (total weight) $|S'|$. So, the same ratio achieved by A on G' is also guaranteed by WA on G.

It is easy to see that this reduction preserves constant approximation ratios. But, it is also easy to see that a ratio $f(\Delta(G))$ (i.e., function of $\Delta(G)$) for MAX INDEPENDENT SET transforms into a ratio $O(f(\Delta(G)w_{\max}))$ (where $w_{\max}$ is the maximum weight-value) and not into $f(\Delta(G))$ for MAX WEIGHTED INDEPENDENT SET. Hence, the reduction does not preserve ratios functions of Δ. The same observation immediately holds for ratios functions of the order of the input-graph.

6. SOME WORDS ON INAPPROXIMALITY

Study of approximability properties of a problem includes two complementary issues: the development of approximation algorithms guaranteeing "good" approximation ratios and the achievement of inapproximability results. The goal of an inapproximability result is to provide answers to a global question, addressed this time not to a single algorithm but to a combinatorial optimization problem Π itself. This stake does not only consist of answering whether the analysis of a specific approximation algorithm for Π is fine or not but, informally, if the algorithm devised is the best possible (with respect to the approximation ratio it guarantees); in other words, it provides answers to questions as: "are there other better algorithms for Π?". Or, more generally, "what is the best approximation ratio that a polynomial algorithm could ever guarantee for Π?". Hence, the goal is to prove that Π is inapproximable within some ratio r unless a very unlikely complexity hypothesis becomes true (the strongest such hypothesis is obviously P = NP).

This type of results is very interesting and adds new insights to computationally hard problems. An important characteristic of complexity theory is that very frequently knowledge is enriched more by impossibility proofs than by possibility ones, even if the latter introduce some pessimism.

"When we exclusively see things from a positive point of view, very frequently we elude fine and efficient approaches, we ignore or we cannot see large avenues that lead to new areas and open new possibilities. When we try to prove the impossible we

have to first apprehend the complete spectrum of the possible" ([60], translation from Greek by the author).

There are three fundamental techniques to prove inapproximability results: the *GAP-reductions*, the *PCP theorem* and the *approximability preserving reductions*. In what follows we shall shortly explain the former of these techniques, that is the older technique for proving such results. For the PCP theorem, the interested reader can referred to [2] as well as to [5, 66, 72].

As we have already mentioned, an inapproximability result for an NPO problem Π consists of showing that if we had an approximation algorithm achieving some approximation ratio r, then this fact would contradict a commonly accepted complexity hypothesis (e.g., P $\neq$ NP). How can we do this? Let us consider the following example showing that MIN COLORING is not approximable within standard-ratio less than $4/3-\varepsilon$, for any $\varepsilon > 0$.

Example 3. Let us revisit the NP-completeness proof for the decision version of MIN COLORING given in [35]. The reduction proposed there constructs, starting from an instance φ of E3SAT[16], a graph G such that if φ is satisfiable then G is 3-colorable (i.e., its vertices can be colored by 3 colors), otherwise G is at least 4-colorable.

Suppose now that there is a polynomial algorithm for MIN COLORING guaranteeing standard-approximation ratio $(4/3)-\varepsilon$, with $\varepsilon > 0$. Run it on the graph G constructed from φ. If φ is not satisfiable, then this algorithm computes a coloring for G using more than 4 colors. On the other hand, if φ is satisfiable (hence G is 3-colorable), then the algorithm produces a coloring using at most $3((4/3)-\varepsilon) < 4$ colors, i.e., a 3-coloring. So, on the hypothesis that a polynomial algorithm for MIN COLORING guaranteeing approximation ratio $(4/3)-\varepsilon$ exists, one can in polynomial time decide if a formula ϕ, instance of E3SAT, is satisfiable or not, contradicting so the NP-completeness of this problem. Hence, MIN COLORING is not $((4/3)-\varepsilon)$-standard-approximable, unless P = NP.

The reduction of Example 6 is a typical example of a GAP-reduction. Via such reductions, one tries to create a gap separating *yes*-instances (i.e., the instances of the problem where answer is *yes*) of a decision problem from *no*-instances (i.e., the instances of the problem where answer is *no*).

More generally, denoting, for some decision problem Π, by O_Π the set of its *yes*-instances, the basic idea of a GAP-reduction is the following.

Consider a decision problem Π that is NP-complete and an NPO problem Π' (suppose w.l.o.g. that goal(Π') = min). If we devise a polynomial reduction from Π to Π' such that there exist $c, r > 1$ for which:

- if I is a *yes*-instance of Π, then $(I') \leq c$,
- if I is a *no*-instance of Π, then $(I') > rc$,

[16] Given a set of m clauses over a set of n variables, SAT consists of finding a model for the conjunction of these clauses, i.e., an assignment of truth values to the variables that simultaneously satisfies all the clauses; E3SAT is the restriction of SAT to clauses with exactly three literals.

then, the reduction above is a GAP-reduction proving that Π' is inapproximable within standard-ratio r.

Indeed, if Π' was approximable within r, then:

- $\forall I \in \mathrm{O}_{\Pi}$ (the set *yes*-instances), $m(f(I), S') \leq rc$;
  for any *no*-instance I, $m(f(I), S') \geq opt(f(I)) > rc$.

Consequently, we would have a separation criterion, i.e., a *gap* (see also Figure 12), that can be checked in polynomial time, between the *yes*- and the *no*-instances of Π. Such a criterion is impossible since Π is NP-complete.

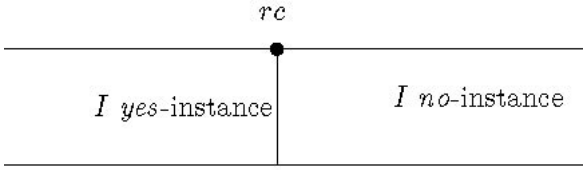

Figure 12: The gap.

How can we extend such results to other problems? Assume, for example, that a GAP-reduction is devised between an NP-complete decision problem Π and an NPO problem Π', deriving that Π' is not approximable within ratio better than r. Suppose also that an approximability preserving reduction (f, g, c) (see Section 5) is devised from Π' to some other NPO problem Π''. Then Π'' *is not approximable within better than* $c^{-1}(r)$.

We conclude this short section with another example of GAP-reduction settling MIN TSP.

Example 4. ([35, 68]).Let us prove that (general) MIN TSP is not approximable within any constant standard-approximation ratio, unless P = NP. Suppose a contrario, that MIN TSP is polynomially approximable within standard-approximation ratio r for some constant r, denote by A an approximation algorithm achieving this ratio and consider a graph $G(V, E)$ of order n, instance of the HAMILTONIAN CYCLE[17] problem. Starting from G construct an instance of MIN TSP as follows:

- complete G in order to build a complete graph K_n;
- for every $e \in E(K_n)$ (the edge-set of K_n) set:

$$d(e) = \begin{cases} 1 & \text{if } e \in E(G) \\ rn & \text{if } e \notin E(G) \end{cases}$$

Consider now the following two cases depending on the fact that G is Hamiltonian (i.e., it has a Hamiltonian cycle) or not:

1. G *is Hamiltonian*; then, $opt(K_n) = n$, since the edges of the Hamiltonian cycle of G exist in K_n and have all weights equal to 1; in this case,

[17] Given a graph G the HAMILTONIAN CYCLE problem consists of deciding if G contains a Hamiltonian tour (see footnote 10); this problem in NP-complete ([35])

Algorithm A will produce a solution (tour) T for MIN TSP of value $m(K_n, T) \leq rn$;

2. G *is Hamiltonian*; then, obviously, no tour in K_n can only use edges weighted by 1 (if such tour existed, it would be a Hamiltonian cycle in G) so, any tour will use at least one edge of weight rn; hence, $opt(K_n) \geq rn + n - 1 > rn$ and, since A will produce something worse than the optimum, $m_{\mathrm{A}}(K_n, T) \geq opt(K_n) > rn$.

We so have a gap between Hamiltonian graphs, deriving MIN TSP solutions of value at most rn, and non-Hamiltonian graphs implying MIN TSP solutions of value greater than rn.

So, on the hypothesis that MIN TSP is approximable within standard-approximation ratio r, one could polynomially solve Hamiltonian cycle as follows:

- starting from an instance G of Hamiltonian cycle, construct instance K_n of MIN TSP as described;
- run A on K_n and if A returns a tour of value at most rn, answer *yes* to Hamiltonian cycle, otherwise answer *no*.

Since everything in the transformation of G into K_n is polynomial and, furthermore, Algorithm A is assumed to be polynomial, the whole algorithm for Hamiltonian cycle is also polynomial, contradicting so the fact that this latter problem is NP-complete.

Inapproximability result of Example 12 can be importantly strengthened. Indeed, observe that the only case where transformation of G into K_n is not polynomial, is when parameter r (the approximation ratio of the polynomial algorithm assumed for MIN TSP) is doubly exponential. Otherwise, even if r is exponential, say of the form $2^{p(n)}$ for any polynomial p of n, the described transformation remains polynomial since any number k can be represented using $O(\log k)$ bits. So, the following result finally holds and concludes the section.

Theorem 9. *Unless* $P = NP$, *MIN TSP cannot be approximately solved within standard-approximation ratios better than* $2^{p(n)}$ *for any polynomial* p.

7. A QUICK "TOUR D'HORIZON" ABOUT COMPLETENESS IN APPROXIMABILITY CLASSES

Given a set C of problems and a reduction R, it is natural to ask if there are problems $\Pi \in C$ such that any problem $\Pi' \in C$, R-reduces to Π. Such "maximal" problems are called in complexity theory *complete problems*. Let C be a class of problems and R be a reduction. A problem $\Pi \in C$ is said to be C-complete (under R-reduction) if for any $\Pi' \in C$, $\Pi' \leq_{\mathrm{R}} \Pi$. A C-complete problem (under reduction R) is then (in the sens of this reduction) a computationally hardest problem for class C. For instance, in the case of NP-completeness, NP-complete problems (under Karp-reduction ([35])) are the hardest problems of NP since if one could polynomially solve just one of them, then one would be able to solve in polynomial time any other problem in NP. Let C

be a class of problems and R a reduction. A problem Π is said to be C-hard (under R-reduction) if for any $\Pi' \in C$, $\Pi' \leq_R \Pi$. In other words, a problem Π is C-complete if and only if $\Pi \in C$ and Π is C-hard.

The general definitions given above can be immediately applied to approximability classes defined in Section 3 in order to produce strong inaprroximability results but also in order to create a structure for these classes. In fact, even if approximability preserving reductions mainly concern the transfer of results among pairs of problems, we can use them as mentioned, in order to complete the structure of approximability classes.

Consider some approximability preserving reduction R and suppose that it preserves membership in, say, PTAS, in other words, if a problem Π R-reduces to Π' and if $\Pi' \in PTAS$, then $\Pi \in PTAS$. Consider now an approximation class that contains PTAS, say APX and assume that the existence of a problem Π that is APX-complete under R-reduction has been proved. If Π admits a polynomial time approximation scheme then, since R-reduction preserves membership in PTAS, one can deduce the existence of polynomial time approximation schemata for any problem that is R-reducible to Π, hence, in particular, for any problem in APX. In other words, by the assumptions just made, we have:

$$\Pi \in PTAS \Rightarrow APX = PTAS$$

Since, under the hypothesis P $\neq$ NP, $PTAS \subset APX$, one can conclude that, under the same hypothesis, $\Pi \notin PTAS$.

The above scheme of reasoning can be generalized for any approximation class. Let C be a class of problems. We say that a reduction R preserves membership in C, if for every pair of problems Π and Π':

$$\Pi \in C \text{ and } \Pi' \leq_R \Pi \Rightarrow \Pi' \in C$$

We then have the following proposition.

Proposition 1. *Let C and C′ be two problem-classes with $C' \not\subset C$. If a problem Π is C-complete under some reduction preserving membership in C′, then $\Pi \notin C'$.*

Obviously, if the strict inclusion of classes is subject to some complexity hypothesis, the conclusion $\Pi \notin \mathbf{C}'$ is subject to the same hypothesis.

The analogy with NP-completeness is immediate. The fundamental property of Karp- (or Turing-) reduction is that it preserves membership in P. Application of Proposition 1 to NP-completeness framework simply says that NP-complete problems can never be in P, unless P = NP.

When the problem of characterizing approximation algorithms for hard optimization problems was tackled, soon the need arose for a suitable notion of reduction that could be applied to optimization problems in order to study their approximability properties.

What is it that makes algorithms for different problems behave in the same way? Is there some stronger kind of reducibility than the simple polynomial reducibility that will explain these results, or are they due to some structural similarity between the problems as we define them? ([46]).

The first answer to the above questions was given by [6, 7] where the notion of *structure preserving reduction* is introduced and for the first time the completeness of

MAX VARIABLE-WSAT (a weighted version of MAX SAT) in the class of NPO problems is proved. Still it took a few more years until suitable kinds of reductions among optimization problems were introduced by [58]. In particular, this paper presented the so-called *strict reduction* and provided the first examples of complete problems (MIN VARIABLE-WSAT, MIN 0-1 LINEAR PROGRAMMING and MIN TSP) under approximation preserving reductions.

After [58] a large variety of approximation preserving reductions have appeared in the literature. The introduction of powerful approximation preserving reductions and the beginning of the structural theory of approximability of optimization problems can be traced back to the fundamental paper by [24] where the first PTAS preserving reductions (the *PTAS-reduction*) is introduced and the first complete problems in APX under such types of reductions are presented. Unfortunately the problem which is proved APX-complete in this paper is quite artificial, MAX VARIABLE-WSAT-B, a version of MAX VARIABLE-WSAT in which, given a constant B, the sum of weights of variables is contained between B and $2B$.

Along a different line of research, during the same years, the study of logical properties of optimization problems has led to the syntactic characterization of an important class of approximable problems, the class Max-SNP (see [62]) strongly based upon characterization of NP by [33]. Completeness in the class Max-SNP has been defined in terms of *L-reductions* and natural complete problems (e.g., MAX 3SAT, MAX 2SAT, MIN VERTEX COVER etc.) have been found. The relevance of such an approach is related to the fact that it is possible to prove that Max-SNP-complete problems do not allow polynomial time approximation schemata, unless P = NP ([2]).

The two approaches have been reconciled by [49], where the closure of syntactically defined classes with respect to an approximation preserving reduction were proved equal to the more familiar computationally defined classes. As a consequence of this result any Max-SNP-completeness result appeared in the literature can be interpreted as an APX-completeness result. In this paper a new type of reduction is introduced, the *E-reduction*. This reduction is fairly powerful since it allows proving that MAX 3SAT is APX-complete. On the other side, it remains somewhat restricted because it does not allow the transformation of PTAS problems (such as KNAPSACK) into problems belonging to APX-PB (the class of problems in APX whose solution-values are bounded by a polynomial in the size of the instance) such as MAX 3SAT. In [49], completeness in approximability classes beyond APX, as Log-APX and Poly-APX has been also tackled and completeness results for subclasses of them (Log-APX-PB and Poly-APX-PB, respectively, where problems have solution-values bounded by a polynomial in the size of the instance) have been proved. The existence of natural complete problems for the whole classes Log-APX and Poly-APX has been proved in [31, 11], respectively, under *FT-reduction* and *MPTAS-reduction*.

The final answer to the problem of finding the suitable kind of reduction (powerful enough to establish completeness results both in NPO and APX) is the *AP-reduction*, introduced by [23].

A large number of other reductions among optimization problems have been introduced throughout the years. Overviews of the world of approximation preserving reductions and completeness is approximability classes can be found in [8, 9, 22, 23].

On the other hand, a structural development analogous to the one that has been carried on for the standard paradigm has been elaborated also for the differential

paradigm that is much younger than the former since it has been defined at the beginning of the 90's by [27]. In [4] and in [11] the approximability classes DAPX and DPTAS are introduced, suitable approximation preserving reductions are defined and complete problems in NPO, 0-DAPX, DAPX, DPTAS and Poly-DAPX, under such kind of reductions, are shown.

Finally, in [31], together with the existence of complete problems for Log-APX and Poly-APX, completeness in class Exp-APX is also proved and tools for proving completeness in classes Log-DAPX and Exp-DAPX, where no natural problems are still proved to belong to, are given.

We conclude this *tour d'horizon*, with a synthesis of the several completeness results briefly presented just above for the combinatorially defined approximability classes seen given in Section 3 (excluding so the synatctically defined class Max-SNP).

For the standard-approximation paradigm:

- NPO-complete: several versions of variable-weighted SAT ([6, 7, 58]);
- Exp-APX-complete: MIN TSP ([31]);
- Poly-APX-complete: MAX INDEPENDENT SET ([11]);
- Log-APX-complete: MIN SET COVER ([31]);
- APX-complete: MAX 3-SAT, MIN VERTEX COVER-B, MAX INDEPENDENT SET-B, MIN TSP WITH EDGE-WEIGHTS 1 AND 2, ... ([24, 23, 49], etc.);
- PTAS-complete: MIN PLANAR VERTEX COVER, MAX PLANAR INDEPENDENT SET ([11]).

For the differential-approximation paradigm:

- 0-DAPX-complete: MIN INDEPENDENT DOMINATING SET, ... ([4]);
- Exp-DAPX-complete: no natural problem is still known to be in Exp-DAPX;
- Poly-DAPX-complete: MAX INDEPENDENT SET, MIN VERTEX COVER, MIN SET COVER, MAX CLIQUE, ... ([31]);
- Log-DAPX-complete: the same as for class Exp-DAPX holds;
- DAPX-complete: MIN VERTEX COVER-B, MAX INDEPENDENT SET-B, ... ([4]), MIN COLORING ([11]);
- DPTAS-complete: MIN PLANAR VERTEX COVER, MAX PLANAR INDEPENDENT SET ([11]), BIN PACKING, ... ([11]).

8. FURTHER REMARKS

Polynomial approximation is a research area in the boundaries several research fields the main among them being combinatorial optimization and theoretical computer science. For more than thirty years, it constitutes a very active research programme that has rallied numerous researchers all over the world. Furthermore, it has inspired several new approaches in both operational research and computer science.

One of these new approaches is a dynamic computation model, called *online approximation*, where the basic hypotheses are that instance to be solved is revealed step-by-step and the algorithm supposed to solve it has to maintain a feasible solution for the part of the instance already revealed. The quality of such an algorithm is measured by means of its *competitive ratio* defined as the ratio of the value of the solution computed during instance's revealing divided by the value of the optimal solution of the whole

instance (called offline solution). Extensions of online computation can deal not only with data arrival but also with data elimination.

Another approach is the so-called *reoptimization*. Here we suppose that we have an optimal solution for an instance (no matter how this solution is produced) and some new data arrive. Can we operate a fast transformation of the solution at hand in order to fit the augmented instance? Is the new solution optimal or not? If not, does it achieve a good approximation ratio?

Notions and tools from polynomial approximation are also used in a relatively new research field that is actually in full expansion: the algorithmic game theory. The so-called *price of anarchy* is fully inspired from polynomial approximation.

What are the mid- and long-term perspectives of this area? Certainly, producing new operational and structural results are such perspectives. For instance, syntactic classes, as class Max-SNP are not yet fully studied in this paradigm. Also, optimal satisfiability problems, central problems in the standard paradigm, deserve further research in differential approximation.

But, to our opinion, major long-term perspective is to match polynomial approximation with exact computation. Indeed, another very active area of combinatorial optimization is the development of exact algorithms for NP-hard problems with non-trivial worst-case running times. For example, it is obvious that an exhaustive method for MAX INDEPENDENT SET will run in time at most 2^n. But can we go faster? Such faster algorithms with improved worst-case complexity have been recently devised for numerous NPO problems. Polynomial approximation and exact computation (with worst case upper time-bounds) can be brought together in several ways. For instance, are we able to produce approximation ratios ``forbidden'' for polynomial algorithms (e.g., constant ratios for MAX INDEPENDENT SET, or ratios smaller than 2 for MIN VERTEX COVER) by exponential or super-polynomial algorithms running in times much lower than those for the exact computation for the corresponding problems? And, in a second time, can we adapt the concept of approximability preserving reductions to fit this new issue?

Acknowledgment. The very useful comments and suggestions of an anonymous referee are gratefully acknowledged.

REFERENCES

[1] Arora, S., “Polynomial time approximation schemes for Euclidean TSP and other geometric problems”, in: *Proc. FOCS'96*, 1996, 2-11.

[2] Arora, S., Lund, C., Motwani, R., Sudan, M., and Szegedy, M., “Proof verification and intractability of approximation problems”, *J. Assoc. Comput. Mach.*, 45 (3) 1998, 501-555.

[3] Aspvall, B., and Stone, R., E., “Khachiyan's linear programming algorithm”, *J. Algorithms*, 1 (1980), 1980, 1-13.

[4] Ausiello, G., Bazgan, C., Demange, M., and Paschos, Th.,V., “Completeness in differential approximation classes” *International Journal of Foundations of Computer Science*, 16 (6) 2005, 1267-1295.

[5] Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., and Protasi, M., *Complexity and Approximation. Combinatorial Optimization Problems and Their Approximability Properties*, Springer-Verlag, Berlin, 1999.

[6] Ausiello, G., D'Atri, A., and Protasi, M., “Structure preserving reductions among convex optimization problems”, *J. Comput. System Sci.*, 21 (1980) 136-153.

[7] Ausiello, G., D'Atri, A., and Protasi, M., "Lattice-theoretical ordering properties for NP-complete optimization problems" *Fundamenta Informaticæ*, 4(1981) 83-94.
[8] Ausiello, G., and Paschos, Th., V., "Reductions, completeness and the hardness of approximability" *European J. Oper. Res.*, 172 (2006) 719-739, Invited review.
[9] Ausiello, G., and Paschos, Th., V., "Reductions that preserve approximability" in: T. F. Gonzalez, (ed), *Handbook of approximation algorithms and metaheuristics*, chapter 15, Chapman & Hall, Boca Raton, (2007) 15-16 .
[10] Bar-Yehuda, R., and Even, S., "A local-ratio theorem for approximating the weighted vertex cover problem" *Ann. Discrete Appl. Math.*, 25 (1985) 27-46.
[11] Bazgan, C., Escoffier, B., and Paschos, Th., V., "Completeness in standard and differential approximation classes: Poly-(D)APX- and (D)PTAS-completeness" *Theoret. Comput. Sci.*, 339 (2005) 272-292.
[12] Bazgan, C., Monnot, J., Paschos, Th., V., and Serriθre, F., "On the differential approximation of min set cover", *Theoret. Comput. Sci.*, 332 (2005) 497-513.
[13] Bazgan, C., and Paschos, Th., V., "Differential approximation for optimal satisfiability and related problems", *European J. Oper. Res.*, 147(2) (2003) 397-404.
[14] Bell, E., T., *The Development of Mathematics*. Dover, 1992.
[15] Bellare, M., Goldreich, O., and Sudan, M., "Free bits and non-approximability - towards tight results", *SIAM J. Comput.*, 27(3) (1998) 804-915.
[16] Berge, C., *Graphs and Hypergraphs,* North Holland, Amsterdam, 1973.
[17] Berman, P., and Karpinski, M., "8/7-approximation algorithm for (1,2)-tsp", in: *Proc. Symposium on Discrete Algorithms, SODA'06*, 2006, 641-648.
[18] Brooks, R., L., "On coloring the nodes of a network", *Math. Proc. Cambridge Philos. Soc.*, 37 (1941) 194-197.
[19] Christofides, N., "Worst-case analysis of a new heuristic for the traveling salesman problem", Technical Report 388, Grad. School of Industrial Administration, CMU, 1976.
[20] Chvátal, V., "A greedy-heuristic for the set covering problem", *Math. Oper. Res.*, 4 (1979) 233-235.
[21] Cook, S., A., "The complexity of theorem-proving procedures", in: *Proc. STOC'71*, pages (1971) 151-158.
[22] Crescenzi, P., "A short guide to approximation preserving reductions", in: *Proc. Conference on Computational Complexity*, 1997, 262-273.
[23] Crescenzi, P., Kann, V., Silvestri, R., and Trevisan, L., "Structure in approximation classes" *SIAM J. Comput.*, 28 (5) (1999) 1759-1782.
[24] Crescenzi, P., and Panconesi, A., "Completeness in approximation classes" *Information and Computation*, 93(2) (1991) 241-262.
[25] Croes, A., "A method for solving traveling-salesman problems". *Oper. Res.*, 5 (1958) 791-812.
[26] Demange, M., De Werra, D., Monnot, J., and Paschos, V., Th., "Time slot scheduling of compatible jobs" *J. Scheduling*, 10 (2007) 111-127.
[27] Demange, M., and Paschos, V., Th., "On an approximation measure founded on the links between optimization and polynomial approximation theory", *Theoret. Comput. Sci.*, 158 (1996) 117-141.
[28] Demange, M., and Paschos, V., Th., "Improved approximations for maximum independent set via approximation chains", *Appl. Math. Lett.*, 10 (3) (1997) 105-110.
[29] Demange, M., and Paschos, V., Th., "Improved approximations for weighted and unweighted graph problems", *Theory of Computing Systems*, 38 (2005) 763-787.
[30] Escoffier, B., and Monnot, J., "Better differential approximation for symmetric TSP", Cahier du LAMSADE 261, LAMSADE, Université Paris-Dauphine, 2007.
[31] Escoffier, B., and Paschos, V., Th., "Completeness in approximation classes beyond APX" *Theoret. Comput. Sci.*, 359 (1-3) (2006) 369-377.

[32] Escoffier, B., and Paschos, V., Th., "A survey on the structure of approximation classes", Cahier du LAMSADE 267, LAMSADE, Université Paris-Dauphine, 2007. Available at http://www.lamsade.dauphine.fr.
[33] Fagin, R., "Generalized first-order spectra and polynomial-time recognizable sets", in: R. M. Karp, (ed), *Complexity of Computations*, American Mathematics Society, 1974, 43-73.
[34] Feige, U., and Kilian, J., "Zero knowledge and the chromatic number" in: *Proc. Conference on Computational Complexity*, 1996, 278-287.
[35] Garey, M., R., and Johnson, D., S., *Computers and Intractability. A Guide to the Theory of NP-completenes,* W. H. Freeman, San Francisco, 1979.
[36] Goldsmidt, O., Hochbaum, D., S., and Yu, G., "A modified greedy heuristic for the set covering problem with improved worst case bound", *Inform. Process. Lett.*, 48 (1993) 305-310, 1993.
[37] Grötschel, M., Lovász, L., and Schrijver, A., "The ellipsoid method and its consequences in combinatorial optimization" *Combinatorica*, 1 (1981) 169-197.
[38] Halldórsson, M., M., "A still better performance guarantee for approximate graph coloring" *Inform. Process. Lett.*, 45 (1) (1993) 19-23.
[39] Halldórsson, M., M., "Approximations of weighted independent set and hereditary subset problems" *J. Graph Algorithms Appli.*, 4 (1) (2000) 1-16.
[40] Halperin, E., "Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs", in: *Proc. Symposium on Discrete Algorithms, SODA'00*, 2000.
[41] Hassin, R., and Lahav, S., "Maximizing the number of unused colors in the vertex coloring problem", *Inform. Process. Lett.*, 52 (1994) 87-90.
[42] Håstad, J., "Clique is hard to approximate within $n^{1-\varepsilon}$", *Acta Mathematica*, 182 (1999) 105-142.
[43] Hochbaum, D., S., "Efficient bounds for the stable set, vertex cover and set packing problems" *Discrete Appl. Math.*, 6 (1983) 243-254.
[44] Hochbaum, D., S., (ed) *Approximation Algorithms for NP-hard Problems*. PWS, Boston, 1997.
[45] Ibarra, O., H., and Kim, C., E., "Fast approximation algorithms for the knapsack and sum of subset problems" *J. Assoc. Comput. Mach.*, 22 (4) (1975) 463-468.
[46] Johnson, D., S., "Approximation algorithms for combinatorial problems" *J. Comput. System Sci.*, 9 (1974) 256-278.
[47] Karp, R., M., "Reducibility among combinatorial problems", in: Miller, R., E., and Thatcher, J., W., (eds), *Complexity of computer computations*, Plenum Press, New York, 1972, 85-103.
[48] Khachian, L., G., "A polynomial algorithm for linear programming" *Dokladi Akademiy Nauk SSSR*, 244 (1979) 1093-1096.
[49] Khanna, S., Motwani, R., Sudan, M., and Vazirani, U., "On syntactic versus computational views of approximability", *SIAM J. Comput.*, 28 (1998) 164-191.
[50] Khot, S., and Regev, O., "Vertex cover might be hard to approximate to within $2-\varepsilon$", in: *Proc. Annual Conference on Computational Complexity, CCC'03*, 2003, 379-386.
[51] Kleene, S., C., *Introduction to Metamathematics*. Van Nostrand, Princeton, NJ, 1952.
[52] Lovász, L., "On the ratio of optimal integral and fractional covers", *Discrete Math.*, 13 (1975) 383-390.
[53] Lovász, L., "Three short proofs in graph theory", *J. Combin. Theory Ser. B*, 19 (1975) 269-271.
[54] Monien, B., and Speckenmeyer, E., "Ramsey numbers and an approximation algorithm for the vertex cover problem", *Acta Informatica*, 22 (1985) 115-123.
[55] Monnot, J., Paschos, V., Th., and Toulouse, S., "Differential approximation results for the traveling salesman problem with distances 1 and 2" *European J. Oper. Res.*, 145 (3) (2002) 557-568.
[56] Monnot, J., Paschos, V., Th., and Toulouse, S., "Approximation algorithms for the traveling salesman problem", *Mathematical Methods of Operations Research*, 57 (1) (2003) 387-405.

[57] Nemhauser, G., L., and Trotter, L., E., "Vertex packings: structural properties and algorithms" *Math. Programming*, 8 (1975) 232-248.
[58] Orponen, P., and Mannila, H., "On approximation preserving reductions: complete problems and robust measures" Technical Report C-1987-28, Dept. of Computer Science, University of Helsinki, Finland, 1987.
[59] Papadimitriou, C., H., *Computational complexity*. Addison-Wesley, 1994.
[60] Papadimitriou, C., H., *Το χαμόγελο του Turing (Turing's Smile)*. Νέα Σύνορα, Athens, 2000. in: Greek. English title: Turing.
[61] Papadimitriou, C., H., and Steiglitz, K., *Combinatorial Optimization: Algorithms and Complexity*. Prentice Hall, New Jersey, 1981.
[62] Papadimitriou, C., H., and Yannakakis, M., "Optimization, approximation and complexity classes" *J. Comput. System Sci.*, 43 (1991) 425-440.
[63] Paschos; V., Th., "A note on the improvement of the maximum independent set's approximation ratio", *Yugoslav Journal of Operations Research*, 5 (1) (1995) 21-26.
[64] Paschos, V., Th., "A survey about how optimal solutions to some covering and packing problems can be approximated", *ACM Comput. Surveys*, 29 (2) (1997) 171-209.
[65] Paschos, V., Th., "Polynomial approximation and graph coloring" *Computing*, 70 (2003) 41-86.
[66] Paschos, V., Th., *Complexité et Approximation Polynomiale*. Hermθs, Paris, 2004.
[67] Raz, R., and Safra, S., "A sub-constant error probability low-degree test and a sub-constant error probability PCP characterization of NP", in: *Proc. STOC'97*, 1997, 475-484.
[68] Sahni, S., and Gonzalez, T., "P-complete approximation problems" *J. Assoc. Comput. Mach.*, 23 (1976) 555-565.
[69] Schrijver, A., *Theory of Linear and Integer Programming,* John Wiley & Sons, New York, 1986.
[70] Simon, H., U., "On approximate solutions for combinatorial optimization problems" *SIAM J. Disc. Math.*, 3(2) (1990) 294-310.
[71] Slavk, P., "A tight analysis of the greedy algorithm for set cover", in: *Proc. STOC'96*, 1996, 435-441.
[72] Vazirani, V., *Approximation Algorithms*, Springer, Berlin, 2001.